Data Preprocessing Practice

September 29, 2025

[51]:
```
import pandas as pd
```

[52]:
```
# Load the dataset from Downloads folder
df = pd.read_csv(r"C:\Users\ASUS\Downloads\employees.csv")
```

[53]:
```
# Show the first 10 rows
df.head(10)
```

[53]:

	EmployeeID	Name	Department	Age	Salary	Experience	JoiningDate	PerformanceScore
0	101	Employee_1	Finance	54	39085.0	8	2017-08-24	93
1	102	Employee_2	HR	47	63837.0	21	2012-10-11	75
2	103	Employee_3	Finance	42	33986.0	5	2021-05-27	63
3	104	Employee_4	IT	48	58977.0	23	2017-07-15	93
4	105	Employee_5	Finance	34	86078.0	1	2022-08-11	63
5	106	Employee_6	Marketing	52	116951.0	4	2015-12-31	82
6	107	Employee_7	IT	30	NaN	1	2023-08-26	60
7	108	Employee_8	IT	45	103634.0	5	2011-12-12	89
8	109	Employee_9	Finance	29	87272.0	2	2022-01-14	99
9	110	Employee_10	HR	49	73332.0	5	2015-12-09	84

[54]:
```
# Display number of rows and columns as (rows,columns)
df.shape
```

[54]:
```
(20, 8)
```

[55]:
```
# Showing all column names and their data types
print(df.dtypes)
```

```
EmployeeID            int64
Name                 object
Department           object
Age                   int64
Salary              float64
Experience            int64
JoiningDate          object
PerformanceScore      int64
dtype: object
```

[56]:
```
# Check for missing values in each column
df.isnull().sum()
```

[56]:
```
EmployeeID          0
Name                0
Department          0
Age                 0
Salary              3
Experience          0
JoiningDate         0
PerformanceScore    0
dtype: int64
```

[57]:
```
# Fill missing salary values with the mean salary of all employees
df['Salary'].fillna(df['Salary'].mean(), inplace=True)
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_11972\621059330.py:2: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Salary'].fillna(df['Salary'].mean(), inplace=True)
```

[58]:
```
# Verify if missing values are filled
df.isnull().sum()
```

[58]:
```
EmployeeID          0
Name                0
```

```
Department          0
Age                 0
Salary              0
Experience          0
JoiningDate         0
PerformanceScore    0
dtype: int64
```

[59]:
```
# Print the Salary column to verify if missing salary values are correctly␣
↪filled
print(df['Salary'])
```

```
0      39085.000000
1      63837.000000
2      33986.000000
3      58977.000000
4      86078.000000
5     116951.000000
6      71472.235294
7     103634.000000
8      87272.000000
9      73332.000000
10     47515.000000
11    102352.000000
12     71472.235294
13     35893.000000
14     71472.235294
15     79347.000000
16     77560.000000
17     51105.000000
18     42408.000000
19    115696.000000
Name: Salary, dtype: float64
```

[60]:
```
# Checking if any cell in department column is missing
df['Department'].isnull().sum()
```

[60]:
```
np.int64(0)
```

[61]:
```
# Drop any rows where department is missing
df = df.dropna(subset=['Department'])
```

[62]:
```
# Check again if any department is missing
df['Department'].isnull().sum()
```

[62]:
```
np.int64(0)
```

[63]:
```
# Average salary of employees in the IT department
avg_it_salary = df[df['Department'] == "IT"]['Salary'].mean()
print("Average salary of employees in IT department:", avg_it_salary)
```

```
Average salary of employees in IT department: 71831.64705882352
```

[64]:
```
# Filter employees with PerformanceScore greater than 90
high_performers = df[df['PerformanceScore'] > 90]

# Display the result
print(high_performers)
```

```
    EmployeeID         Name Department  Age     Salary  Experience JoiningDate  \
0          101   Employee_1    Finance   54    39085.0           8  2017-08-24
3          104   Employee_4         IT   48    58977.0          23  2017-07-15
8          109   Employee_9    Finance   29    87272.0           2  2022-01-14
11         112  Employee_12         HR   36   102352.0           3  2022-03-14
15         116  Employee_16    Finance   38    79347.0          19  2014-04-20

    PerformanceScore
0                 93
3                 93
8                 99
11                95
15                93
```

[65]:
```
# Find the index of the employee with the highest salary
highest_paid_index = df['Salary'].idxmax()

# Get the details of the highest paid employee
highest_paid_employee = df.loc[highest_paid_index]

print(highest_paid_employee)
```

```
EmployeeID                106
Name                Employee_6
Department           Marketing
Age                         52
Salary                116951.0
Experience                   4
JoiningDate         2015-12-31
PerformanceScore            82
Name: 5, dtype: object
```

[66]:
```
# Group by department and calculating average age
avg_age_by_department = df.groupby('Department')['Age'].mean()

print(avg_age_by_department)
```

```
Department
Finance      37.714286
HR           37.833333
IT           45.600000
Marketing    53.500000
Name: Age, dtype: float64
```

[67]:
```python
# Create Bonus column as 10% of Salary
df['Bonus'] = df['Salary'] * 0.10

# Print the complete dataset
print(df.to_string())
```

```
    EmployeeID         Name Department  Age         Salary  Experience
JoiningDate  PerformanceScore         Bonus
0          101   Employee_1    Finance   54   39085.000000           8
2017-08-24                93   3908.500000
1          102   Employee_2         HR   47   63837.000000          21
2012-10-11                75   6383.700000
2          103   Employee_3    Finance   42   33986.000000           5
2021-05-27                63   3398.600000
3          104   Employee_4         IT   48   58977.000000          23
2017-07-15                93   5897.700000
4          105   Employee_5    Finance   34   86078.000000           1
2022-08-11                63   8607.800000
5          106   Employee_6  Marketing   52  116951.000000           4
2015-12-31                82  11695.100000
6          107   Employee_7         IT   30   71472.235294           1
2023-08-26                60   7147.223529
7          108   Employee_8         IT   45  103634.000000           5
2011-12-12                89  10363.400000
8          109   Employee_9    Finance   29   87272.000000           2
2022-01-14                99   8727.200000
9          110  Employee_10         HR   49   73332.000000           5
2015-12-09                84   7333.200000
10         111  Employee_11         IT   52   47515.000000           7
2016-10-24                67   4751.500000
11         112  Employee_12         HR   36  102352.000000           3
2022-03-14                95  10235.200000
12         113  Employee_13  Marketing   55   71472.235294          28
2013-06-20                86   7147.223529
13         114  Employee_14         HR   26   35893.000000           6
2015-11-07                84   3589.300000
14         115  Employee_15    Finance   38   71472.235294          10
2017-11-29                60   7147.223529
15         116  Employee_16    Finance   38   79347.000000          19
2014-04-20                93   7934.700000
16         117  Employee_17         IT   53   77560.000000           4
```

```
2023-09-07                82   7756.000000
17          118  Employee_18          HR   27   51105.000000          16
2023-08-24                86   5110.500000
18          119  Employee_19          HR   42   42408.000000           3
2020-04-03                58   4240.800000
19          120  Employee_20     Finance   29  115696.000000           7
2010-04-16                50  11569.600000
```

[68]:
```
# Sort by Experience in ascending order
df_sorted = df.sort_values(by='Experience', ascending=True)

# Display the sorted DataFrame
print(df_sorted)
```

```
    EmployeeID         Name Department  Age         Salary  Experience  \
6          107   Employee_7         IT   30   71472.235294           1
4          105   Employee_5    Finance   34   86078.000000           1
8          109   Employee_9    Finance   29   87272.000000           2
11         112  Employee_12         HR   36  102352.000000           3
18         119  Employee_19         HR   42   42408.000000           3
5          106   Employee_6  Marketing   52  116951.000000           4
16         117  Employee_17         IT   53   77560.000000           4
9          110  Employee_10         HR   49   73332.000000           5
7          108   Employee_8         IT   45  103634.000000           5
2          103   Employee_3    Finance   42   33986.000000           5
13         114  Employee_14         HR   26   35893.000000           6
10         111  Employee_11         IT   52   47515.000000           7
19         120  Employee_20    Finance   29  115696.000000           7
0          101   Employee_1    Finance   54   39085.000000           8
14         115  Employee_15    Finance   38   71472.235294          10
17         118  Employee_18         HR   27   51105.000000          16
15         116  Employee_16    Finance   38   79347.000000          19
1          102   Employee_2         HR   47   63837.000000          21
3          104   Employee_4         IT   48   58977.000000          23
12         113  Employee_13  Marketing   55   71472.235294          28

   JoiningDate  PerformanceScore         Bonus
6   2023-08-26                60   7147.223529
4   2022-08-11                63   8607.800000
8   2022-01-14                99   8727.200000
11  2022-03-14                95  10235.200000
18  2020-04-03                58   4240.800000
5   2015-12-31                82  11695.100000
16  2023-09-07                82   7756.000000
9   2015-12-09                84   7333.200000
7   2011-12-12                89  10363.400000
2   2021-05-27                63   3398.600000
13  2015-11-07                84   3589.300000
```

```
10  2016-10-24                67   4751.500000
19  2010-04-16                50  11569.600000
0   2017-08-24                93   3908.500000
14  2017-11-29                60   7147.223529
17  2023-08-24                86   5110.500000
15  2014-04-20                93   7934.700000
1   2012-10-11                75   6383.700000
3   2017-07-15                93   5897.700000
12  2013-06-20                86   7147.223529
```

[69]:
```
# Sort by Experience in descending order, inplace=True updates df directly
df_sorted_desc = df.sort_values(by='Experience', ascending=False, inplace=True)

# Display the sorted DataFrame
print(df_sorted_desc)
```

```
None
```

[70]:
```
# Get top 5 employees by PerformanceScore
top5_performers = df.nlargest(5, 'PerformanceScore')

# Display the result
print(top5_performers)
```

```
    EmployeeID         Name Department  Age    Salary  Experience JoiningDate  \
8          109   Employee_9    Finance   29   87272.0           2  2022-01-14
11         112  Employee_12         HR   36  102352.0           3  2022-03-14
3          104   Employee_4         IT   48   58977.0          23  2017-07-15
15         116  Employee_16    Finance   38   79347.0          19  2014-04-20
0          101   Employee_1    Finance   54   39085.0           8  2017-08-24

    PerformanceScore    Bonus
8                 99   8727.2
11                95  10235.2
3                 93   5897.7
15                93   7934.7
0                 93   3908.5
```

[71]:
```
# Count number of employees in each department
department_counts = df['Department'].value_counts()

print(department_counts)
```

```
Department
Finance      7
HR           6
IT           5
Marketing    2
Name: count, dtype: int64
```

[72]:
```
# Exporting the cleaned and updated dataset to a new CSV file named␣
↪employees_cleaned.csv
df.to_csv("employees_cleaned.csv", index=False)
```

[73]:
```
import os
print(os.getcwd())
```

```
C:\Users\ASUS
```

[ ]:
```
!jupyter nbconvert --to pdf "Data Preprocessing Practice.ipynb" --output "C:/
↪Users/ASUS/Downloads/data_preprocessing_practice.pdf"
```

[ ]:
```

```